Aim: Find the Outlier from the given data set using trimming and capping methods.

In [6]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:
```
df = pd.read_csv('placement.csv')
df
```

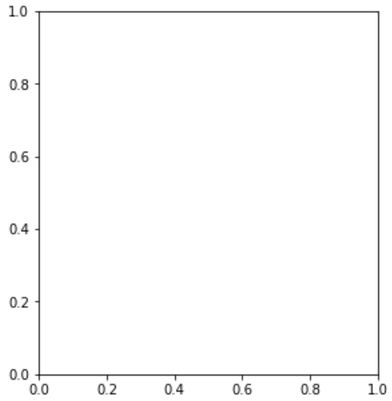
Out[7]:

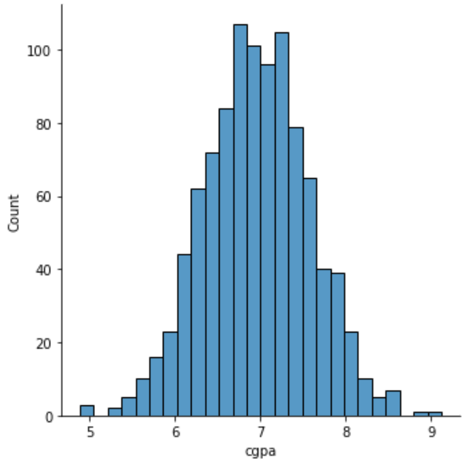
	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

1000 rows × 3 columns

In [14]:
```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.displot(df['cgpa'])
```

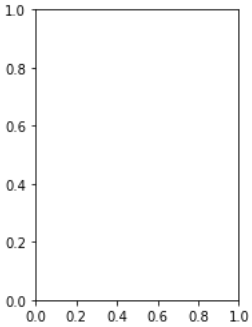
Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x200fe1a2610>
```

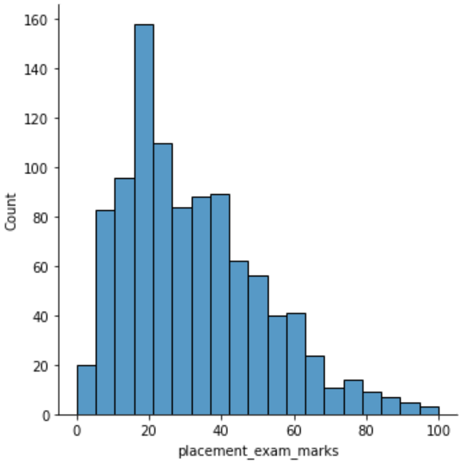




In [10]:
```
plt.subplot(1,2,2)
sns.displot(df['placement_exam_marks'])
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x200f5e9e520>
```





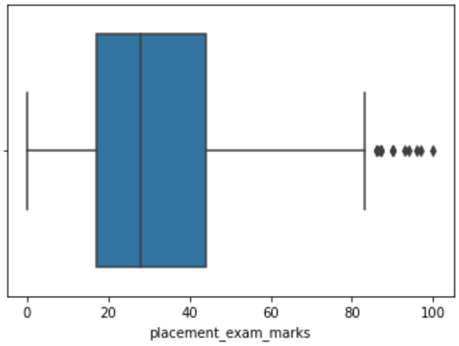
In [15]:
```
df['placement_exam_marks'].describe()
```

Out[15]:
```
count    1000.000000
mean       32.225000
std        19.130822
min         0.000000
25%        17.000000
50%        28.000000
75%        44.000000
max       100.000000
Name: placement_exam_marks, dtype: float64
```

In [17]:
```
sns.boxplot(df['placement_exam_marks'])
```

```
C:\Users\User39\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:
```
<AxesSubplot:xlabel='placement_exam_marks'>
```



In [18]:

```
# finding highest boundries values
print('Highest Boundary value of Cgpa',df['cgpa'].mean() + 3*df['cgpa'].std())
```

```
Highest Boundary value of Cgpa 8.808933625397177
```

In [19]:

```
# Finding lowest boundries value
print('Lowest Boundary value of Cgpa',df['cgpa'].mean() - 3*df['cgpa'].std())
```

```
Lowest Boundary value of Cgpa 5.113546374602842
```

In [20]:

```
# finding outliers
df[(df['cgpa']>8.80)| (df['cgpa']<5.11)]
```

Out[20]:

	cgpa	placement_exam_marks	placed
485	4.92	44.0	1
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
999	4.90	10.0	1

Trimming

In [21]:

```
df.shape
```

Out[21]:

```
(1000, 3)
```

In [27]:

```
new_df = df[(df['cgpa']<8.80) & (df['cgpa']>5.11)]
new_df
```

Out[27]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
...	...	...	...
991	7.04	57.0	0
992	6.26	12.0	0
993	6.73	21.0	1
994	6.48	63.0	0
998	8.62	46.0	1

995 rows × 3 columns

In [26]:

```
new_df.shape
```

Out[26]:

```
(995, 3)
```

Z Score

In [31]:
```
df['cgpa_score'] = (df['cgpa'] - df['cgpa'].mean())/df['cgpa'].std()
df
```

Out[31]:

	cgpa	placement_exam_marks	placed	cgpa_score
0	7.19	26.0	1	0.371425
1	7.46	38.0	1	0.809810
2	7.54	40.0	1	0.939701
3	6.42	8.0	1	-0.878782
4	7.23	17.0	0	0.436371
...	...	...	...	...
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062
997	4.89	34.0	0	-3.362960
998	8.62	46.0	1	2.693239
999	4.90	10.0	1	-3.346724

1000 rows × 4 columns

In [32]:
```
df.describe()
```

Out[32]:

	cgpa	placement_exam_marks	placed	cgpa_score
count	1000.000000	1000.000000	1000.000000	1.000000e+03
mean	6.961240	32.225000	0.489000	-1.600275e-14
std	0.615898	19.130822	0.500129	1.000000e+00
min	4.890000	0.000000	0.000000	-3.362960e+00
25%	6.550000	17.000000	0.000000	-6.677081e-01
50%	6.960000	28.000000	0.000000	-2.013321e-03
75%	7.370000	44.000000	1.000000	6.636815e-01
max	9.120000	100.000000	1.000000	3.505062e+00

In [33]:
```
df['cgpa_score'].describe()
```

Out[33]:
```
count    1.000000e+03
mean    -1.600275e-14
std      1.000000e+00
min     -3.362960e+00
25%     -6.677081e-01
50%     -2.013321e-03
75%      6.636815e-01
max      3.505062e+00
Name: cgpa_score, dtype: float64
```

In [35]:
```
df[df['cgpa_score']>3]
```

Out[35]:

	cgpa	placement_exam_marks	placed	cgpa_score
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062

In [37]:
```
df[df['cgpa_score']< -3]
```

Out[37]:

	cgpa	placement_exam_marks	placed	cgpa_score
485	4.92	44.0	1	-3.314251
997	4.89	34.0	0	-3.362960
999	4.90	10.0	1	-3.346724

In [44]:
```
new_df = df[(df['cgpa_score']<3) & (df['cgpa_score']>-3)]
new_df.shape
```

Out[44]:
```
(995, 4)
```

Capping

In [45]:
```
upper_limit = df['cgpa'].mean() + 3*df['cgpa'].std()
lower_limit = df['cgpa'].mean() - 3*df['cgpa'].std()
lower_limit
```

Out[45]:
```
5.113546374602842
```

In [46]:
```
df['cgpa_cap'] = np.where(
    df['cgpa']>upper_limit,
    upper_limit,
    np.where(
    df['cgpa']<lower_limit,
        lower_limit,df['cgpa']


    )


)
```

In [48]:
```
df.describe()
```

Out[48]:

	cgpa	placement_exam_marks	placed	cgpa_score	cgpa_cap
count	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000
mean	6.961240	32.225000	0.489000	-1.600275e-14	6.961499
std	0.615898	19.130822	0.500129	1.000000e+00	0.612688
min	4.890000	0.000000	0.000000	-3.362960e+00	5.113546
25%	6.550000	17.000000	0.000000	-6.677081e-01	6.550000
50%	6.960000	28.000000	0.000000	-2.013321e-03	6.960000
75%	7.370000	44.000000	1.000000	6.636815e-01	7.370000
max	9.120000	100.000000	1.000000	3.505062e+00	8.808934

Conclusion: Detected Outlier using Trimming and Capping when the data is normally distributed.

In [ ]: